

AMENDMENTS TO THE SPECIFICATION

Please amend paragraph [0005] as follows:

[0005] The task of developing such algorithms is not straightforward. In addition to the requirements discussed above, fundamental requirements of congestion control algorithms include efficient use of bandwidth, fair allocation of bandwidth among sources and that the network should be responsive rapidly to reallocate bandwidth as required. These requirements must be met while respecting key constraints including ~~decentralised~~ decentralized design (TCP sources have restricted information available to them), scalability (the qualitative properties of networks employing congestion control algorithms should be independent of the size of the network and of a wide variety of network conditions) and suitable backward compatibility with conventional TCP sources.

Please amend paragraph [0006] as follows

[0006] To place the disclosed methods and systems in context, the existing TCP network model will now be described. The TCP standard defines a variable *cwnd* that is called the “congestion window”. This variable determines the number of unacknowledged packets that can be in transit at any time; that is, the number of packets in the ‘pipe’ formed by the links and buffers in a transmission path. When the window size is exhausted, the source must wait for an acknowledgement (ACK) before sending a new packet. Congestion control is achieved by dynamically varying *cwnd* according to an additive-increase multiplicative-decrease (AIMD) law. The aim is for a source to probe the network gently for spare capacity and back-off its send rate rapidly when congestion is detected. A cycle that involves an ~~increases~~ increase and a subsequent back-off is termed a “congestion epoch”. The second part is referred to as the “recovery phase”.

Please amend paragraph [0010] as follows

[0010] The current TCP congestion control algorithm described above may be inefficient on modern high-speed and long distance networks. On such links, the window sizes can be very large (perhaps tens of thousands of packets). Following a congestion event, the window size is halved and subsequently only increased by one packet per round-trip time. Thus, it can take a substantial time for the window size to recover, during which time the

send rate is well below the capacity of the link. One possible solution is to simply make the TCP increase parameter α_i larger, thereby decreasing the time to recover following a congestion event and improving the responsiveness of the TCP flow. Unfortunately, this direct solution is inadmissible because of the requirement on lower speed networks for backward compatibility and fairness with existing TCP traffic. The requirement is thus for α_i to be large in high-speed networks but unity in low-speed ones, naturally leading to consideration of some form of mode switch. However, mode switching creates the potential for introducing undesirable dynamic behaviours ~~behaviours~~ in otherwise well behaved systems and any re-design of TCP therefore needs to be carried out with due regard to such issues.

Please amend paragraph [0014] as follows

[0014] The value of may α_i increase at a fixed time after the start of each congestion epoch, for example as a fixed multiple of the round-trip time for a data packet to travel over the network link. As a development of this arrangement, the value of α_i may ~~increases~~ increase at a plurality of fixed times after the start of each congestion epoch. In this case, each fixed time may be calculated as a respective fixed multiple of the round-trip time for a data packet to travel over the network link.

Please amend paragraph [0036] as follows

[0036] The strategy is motivated by and realizes several design criteria as will now be described.

- Sources deploying H-TCP should behave as a normal TCP-source when operating on low-speed communication links. Such ~~behaviour~~ ~~behavior~~ is guaranteed by (4) since the protocol tests the low-speed or high-speed status of the network every congestion epoch.
- Normal AIMD sources competing for bandwidth should be guaranteed some (small) share of the available bandwidth.
- H-TCP sources competing against each other should receive a fair share of the bandwidth. This is guaranteed using the symmetry arguments presented above.

H-TCP sources should be responsive. Again, this is guaranteed using symmetry and an appropriate value of β_i combined with a value of α_i that ensures that the congestion epochs are of suitably short duration.

Please amend paragraph [0044] as follows

[0044] In this example, two H-TCP flows show rapid convergence to fairness. The second flow experiences a drop early in slow-start, ~~focussing~~ focusing attention on the responsiveness of the congestion avoidance algorithm (NS simulation, network parameters: 500Mb bottleneck link, 100ms delay, queue 500 packets; TCP parameters: $\alpha^L = 1$; $\alpha^H = 20$; $\beta_i = 0.5$; $\Delta^L = 19$ corresponding to a window size threshold of 38 packets).

Please amend paragraph [0051] as follows

[0051] The ratio $\frac{RTT_{min,i}}{RTT_{max,i}}$ may approach unity on under-provisioned links. However values of β_i close to unity will give slow convergence after a disturbance (e.g. traffic joining or leaving the route associated with the link, see examples below). It follows that a further adaptive mechanism is desirable which continuously adjusts the trade-off between network responsiveness and efficient link ~~utilization~~ utilization. This requires a network quantity that changes predictably during disturbances and that can be used to trigger an adaptive reset. One variable that does this is the minimum of the mean inter-packet time ($IPT_{min,i}$), where the mean is taken over a round-trip time period. Another variable is the mean throughput. The $IPT_{min,i}$ is a measure of the link bandwidth allocated to a particular flow. This in turn is determined by the link service rate B (which is assumed to be constant), the number of flows and the distribution of bandwidth among the flows. Thus as new flows join, the $IPT_{min,i}$ for an existing flow can be expected to increase. On the other hand, the value of $IPT_{min,i}$ will decrease when the traffic decreases. Thus, by monitoring $IPT_{min,i}$ for changes it is possible to detect points at which the flows need to be adjusted and reset β_i to some suitable low value for a time.